

DISTRIBUTED SNAPSHOT PROCESS

Inventors:
G. Paul Koning
Peter C. Hayden
Paula Long
Hsin H. Lee

5

Reference to Related Application

10 This application claims priority to US Provisional Application USSN 60/441,810
filed 1/21/003 and naming G. Paul Koning, among others, as an inventor, the contents of
which are incorporated by reference.

Field of the Invention

The invention relates to data storage and in particular to systems and methods for
15 backing up data blocks across a plurality of servers.

Background of the Invention

It is desirable during the operation of a mass storage system to periodically gather
information about how the data is stored on the system and from time-to-time to make a
backup copy of the stored data. Gathering such information can be beneficial for a
20 number of reasons, including for recovery in the event of a non-recoverable failure.

Backing up a mass storage system is typically done by reading the data stored on the

mass storage system and writing it to a magnetic tape to create an archive copy of the stored data.

However, generating such archival copies can be burdensome. Many prior art backup methods require that the system be removed from ongoing (online) operations to assure the integrity and consistency of the backup copy. This is because normal backup techniques either copy the blocks from the mass storage system sequentially to a linear-access tape, or walk through the file system on the mass storage system, starting with the first block of the first file in the first directory and proceeding in order to the last block of the last file of the last directory. In either case, the backup process is unaware of updates being performed as data is being written to tape.

Thus, to permit continued, online operations while performing backup operations generates inconsistencies if the data is modified as the backup operation proceeds. Removing the storage system from continued storage operations eliminates the risk of inconsistencies arising during the system operations. However, backup operations can be time consuming therefore making removal of the system from operations undesirable.

One approach to addressing this problem, has been by creating a mirror, or identical copy, of one disk's data. When a backup operation is required, the mirror disk may be used as a static image for a storage. When the static image is no longer necessary (for example, when the tape backup has been completed), the two disks are resynchronized, by copying any changes made during the time mirroring was not active to the mirror

disk, and mirroring is resumed.

Although, mirroring works well, it requires that the data stored on the system be captured accurately. Today however, new distributed storage systems are being developed that avoid the use of a centralized storage control system. These distributed
5 systems capture the benefits of the more flexible and scalable distributed server architectures. Although very exciting, these storage systems present challenges that prior art storage systems do not. One such challenge is the ability to generate reliable and trustworthy archive copies of a data volume that has been distributed across a plurality of independently operating servers.

10 Accordingly, there is a need in the art for a distributed storage system that can provide reliable snapshots of the data volumes that are being maintained across the different server in the system.

Summary of the Invention

15 The systems and methods described herein, include server systems that comprise a group of servers that support a service or resource that has been partitioned across the individual servers of the group. In one application, the systems and methods provide a partitioned storage service for providing storage services to a plurality of clients. In this embodiment, a data volume may be partitioned across a plurality of servers, with each
20 server being responsible for a portion of the data volume. In such a partitioned storage system, the storage “volumes” may be understood as analogous to disk drives in a conventional storage system. However, in the partitioned service, the data volumes have

been spread over several servers, with each server owning a portion of the data within the volume.

For the purpose of fault tolerance, data back-up, and other benefits, the partitioned storage services described herein provide a storage administrator with a snapshot process and system that creates a copy of the state of the storage volume. Typically, the snapshot process results in the creation of a second storage volume, which acts as an archive of the state of the storage system at a given time. Storage administrators may employ this archive as a recovery tool in the case that the original storage volumes fails at a later time, a backup tool for off-line backups, or for any other suitable reason.

Further features and advantages of the invention will be apparent from the following description of preferred embodiments and from the claims.

Brief Description of the Drawings

The following figures depict certain illustrative embodiments of the invention in which like reference numerals refer to like elements. These depicted embodiments are to be understood as illustrative of the invention and not as limiting in any way.

FIG. 1 is a schematic diagram of a client-server architecture with servers organized in server groups;

FIG. 2 is a schematic diagram of the server groups as seen by a client;

FIG. 3 shows details of the information flow between the client and the servers of a group;

FIG. 4 is a process flow diagram for retrieving resources in a partitioned resource environment;

FIG. 5 depicts in more detail and as a functional block diagram one embodiment of a system according to the invention;

FIG. 6 depicts an example of a routing table suitable for use with the system of FIG. 1.

5 FIG. 7 depicts one process for generating a snapshot of a storage volume supported by the system of FIG. 1; and

FIG. 8 depicts an alternate process for generating a snapshot of a storage volume.

Detailed Description of Certain Illustrated Embodiments

The systems and methods described herein include systems for organizing and
10 managing resources that have been distributed over a plurality of servers on a data network. More particularly, the systems and methods described herein include systems and methods for providing archiving processes for a partitioned storage service. However, it will be understood by those of skill in the art that the invention may be employed with applications and services other than storage applications including
15 although are not limited to, distributed file systems, and database systems. Moreover, it will be understood by those of ordinary skill in the art that the systems and methods described herein are merely exemplary of the kinds of systems and methods that may be achieved through the invention and that these exemplary embodiments may be modified, supplemented and amended as appropriate for the application at hand.

Referring first to FIG. 1 one embodiment of a system according to the invention is depicted. As show in FIG. 1, one or several clients 12 are connected, for example via a network 14, such as the Internet, an intranet, a WAN or LAN, or by direct connection, to servers 161, 162, and 163 that are part of a server group 16.

5 The depicted clients 12 can be any suitable computer system such as a PC workstation, a handheld computing device, a wireless communication device, or any other such device, equipped with a network client program capable of accessing and interacting with the server group 16 to exchange information with the server group 16. Optionally, the client 12 and the server group 16 rely on an unsecured communication
10 path for accessing services at the remote server group 16. To add security to such a communication path, the client 12 and the server group 16 may employ a security system, such as any of the conventional security systems that have been developed to provide a secure channel for transmitting data over a network. One such system is the Netscape secured socket layer (SSL) security mechanism that provides a trusted path
15 between a client and a server.

FIG. 1 further depicts that the client 12 may communicate with a plurality of servers, 161, 162 and 163. The servers 161, 162 and 163 employed by the system 10 may be conventional, commercially available server hardware platforms, such as a Sun Sparc™ systems running a version of the Unix operating system. However any suitable
20 data processing platform may be employed. Moreover, it will be understood that one or

more of the servers 161, 162 or 163 may comprise a storage device, such as a tape library, or other device, that is networked with the other servers and clients through network 14.

Each server 161, 162 and 163 may include software components for carrying out the
5 operation and the transactions described herein, and the software architecture of the servers 161, 162 and 163 may vary according to the application. In certain embodiments, the servers 161, 162 and 163 may employ a software architecture that builds certain of the processes described below into the server's operating system, into device drivers, into application level programs, or into a software process that operates
10 on a peripheral device, such as a tape library, a RAID storage system or some other device. In any case, it will be understood by those of ordinary skill in the art, that the systems and methods described herein may be realized through many different embodiments, and practices, and that the particular embodiment and practice employed will vary as a function of the application of interest and all these embodiments and
15 practices fall within the scope hereof.

In operation, the clients 12 will have need of the resources partitioned across the server group 16. Accordingly, each of the clients 12 will send requests to the server group 16. In a typical operation, a client 12 will contact one of the servers, for example server 161, in the group 16 to access a resource, such as a data block, page, file, database
20 table, application, or other resource. The contacted server 161 itself may not hold or

have control over the requested resource. However, in a preferred embodiment, the server group 16 is configured to make all the partitioned resources available to the client 12 regardless of the server that initially receives the request. For illustration, the diagram shows two resources, one resource 18 that is partitioned over all three servers, servers 161, 162, 163, and another resource 17 that is partitioned over two of the three servers. In the exemplary application of the system 10 being a block data storage system, each resource 18 and 17 may represent a partitioned block data volume.

The depicted server group 16 therefore provides a block data storage service that may operate as a storage area network (SAN) comprised of a plurality of servers, servers 161, 162 and 163. Each of the servers 161, 162 and 163 may support one or more portions of the partitioned block data volumes 18 and 17. In the depicted system 10, there are two data volumes and three servers, however there is no specific limit on the number of servers. Similarly, there is no specific limit on the number of resources or data volumes. Moreover, each data volume or several data volumes may be contained entirely on a single server, or may be partitioned over several servers, either all of the servers in the server group, or a subset of the server group. In practice, there may of course be limits due to implementation considerations, for example the amount of memory available in the servers 161, 162 and 163 or the computational limitations of the servers 161, 162 and 163. The grouping itself, i.e., deciding which servers will comprise a group, may in one practice involve an administrative decision. In a typical scenario, a group might at first contain only a few servers, or perhaps only one. The system

administrator would add servers to a group as needed to obtain the level of performance required. Increasing servers creates more space (memory, disk storage) for resources that are stored, more CPU processing capacity to act on the client requests, and more network capacity (network interfaces) to carry the requests and responses from and to the clients. It will be appreciated by those of skill in the art that the systems described herein are readily scaled to address increased client demands by adding additional servers into the group 16.

Referring now to FIG. 2, a client 12 connecting to a server 161 (FIG. 1) will see the server group 16 as if the group were a single server having multiple IP addresses. The client 12 is not aware that the server group 16 is constructed out of a potentially large number of servers 161, 162, 163, nor is it aware of the partitioning of the block data volumes 17, 18 over the several servers 161, 162, 163. As a result, the number of servers and the manner in which resources are partitioned among the servers may be changed without affecting the network environment seen by the client 12.

FIG. 3 shows the resource 18 of FIG. 2 as being partitioned across servers 161, 162 and 163. In the partitioned server group 16, any data volume may be spread over any number of servers within the group 16. As seen in FIGS. 1 and 2, one volume 17 (Resource 1) may be spread over servers 162, 163, whereas another volume 18 (Resource 2) may be spread over servers 161, 162, 163. Advantageously, the respective volumes may be arranged in fixed-size groups of blocks, also referred to as "pages",

wherein an exemplary page contains 8192 blocks. Other suitable page sizes may be employed. In an exemplary embodiment, each server in the group 16 contains a routing table 165 for each volume, with the routing table 165 identifying the server on which a specific page of a specific volume can be found. For example, when the server 161
5 receives a request from a client 12 for volume 3, block 93847, the server 161 calculates the page number (page 11 in this example for the page size of 8192) and looks up in the routing table 165 the location or number of the server that contains page 11. If server 163 contains page 11, the request is forwarded to server 163, which reads the data and returns the data to the server 161. Server 161 then send the requested data to the client
10 12. The response may be returned to the client 12 via the same server 161 that received the request from the client 12.

Accordingly, it is immaterial to the client 12 as to which server 161, 162, 163 has the resource of interest to the client 12. As described above, the servers 162, 162 and 163 will employ the routing tables to service the client request, and the client 12 need
15 not know ahead of time which server is associated with the requested resource. This allows portions of the resource to exist at different servers. It also allows resources, or portions thereof, to be moved while the client 12 is connected to the server group 16. Upon moving a resource, the routing tables 165 are updated as necessary and subsequent client requests will be forwarded to the server now responsible for handling that request.
20 At least within a resource 17 or 18, the routing tables 165 may be identical.

FIG. 4 depicts an exemplary request handling process 40 for handling client requests in a partitioned server environment. The request handling process 40 begins at 41 by receiving a request for a resource, such as a file or blocks of a file, at 42. The request handling process 40 examines the routing table, in operation 43, to determine at 5 which server the requested resource is located. If the requested resource is present at the initial server, the initial server returns the requested resource to the client 12, at 48, and the process 40 terminates at 49. Conversely, if the requested resource is not present at the initial server, the server will use the data from the routing table to determine which server actually holds the resource requested by the client, operation 45. The request is 10 then forwarded to the server that holds the requested resource, operation 46, which returns the requested resource to the initial server, operation 48. The process 40 then goes to 48 as before, to have the initial server forward the requested resource to the client 12, and the process 40 terminates, at 49.

Accordingly, one of ordinary skill in the art will see that the system and methods 15 described herein are capable of partitioning one or more resources over a plurality of servers thereby providing a server group capable of handling requests from multiple clients. Additionally, the above description illustrates that the systems and methods described herein can redistribute or repartition the resource to change how portions of the resource are distributed or spread across the server group. The resources spread over 20 the several servers can be directories, individual files within a directory, blocks within a

file or any combination thereof. Other partitioned services may be realized. For example, it may be possible to partition a database in an analogous fashion or to provide a distributed file system, or a distributed or partitioned server that supports applications being delivered over the Internet. In general, the approach may be applied to any service
5 where a client request can be interpreted as a request for a piece of the total resource.

Turning now to FIG. 5, one particular embodiment of the system 10 is depicted wherein the system is capable of generating a distributed snapshot of the storage volume 18 partitioned across the servers 161, 162 and 163. Specifically, FIG. 5 depicts the system 10 wherein the clients 12 communicate with the server group 16. The server
10 group 16 includes three servers, server 161, 162 and 163. In the embodiment of FIG. 5 the servers 161, 162 and 163 are equivalent servers, in that each of the servers will provide substantially the same resource to the same request from a client. As such, from the perspective of the clients 12, the server group 16 appears to be a single server system that provides multiple network or IP addresses for communicating with clients 12. Each
15 server includes a routing table, depicted as routing tables 20A, 20B and 20C, and a snapshot process 22A, 22B and 22C respectively. Further, and for the purpose of illustration only, the FIG. 5 represents the resources as pages of data 28 that may be copied to generate a second storage volume that is an image of the original storage volume 18.

As shown in FIG. 5, each of the routing tables 20A, 20B and 20C are capable of communicating with each other for the purpose of sharing information. As described above, the routing tables may track which of the individual equivalent servers is responsible for a particular resource maintained by the server group 16. In the

5 embodiment shown in FIG. 5 the server group 16 may form a SAN wherein each of the equivalent servers 161, 162 and 163 has an individual IP address that may be employed by a client 12 for accessing that particular equivalent server on the SAN. As further described above, each of the equivalent servers 161, 162 and 163 may be capable of providing the same response to the same request from a client 12. To that end, the

10 routing tables 20A, 20B and 20C of the individual equivalent 161, 162 and 163 coordinate with each other to provide a global database of the different resources, and the specific equivalent servers that are responsible for those resources.

As depicted in FIG. 6, each routing table includes an identifier (Server ID) for each

15 of the equivalent servers 161, 162 and 163 that support the partitioned data block storage service. Additionally, each of the routing tables includes a table that identifies those data pages associated with each of the respective equivalent servers. In the embodiment depicted by FIG. 6, the equivalent servers support two partitioned volumes. A first one of the volumes, Volume 18, is distributed or partitioned across all three equivalent

servers 161, 162 and 163. The second partitioned volume, Volume 17, is partitioned across two of the equivalent servers, servers 162 and 163 respectively.

Returning now to Figure 5, it can be seen that each of the equivalent servers 161, 162 and 163 includes a snapshot process 22a, 22b and 22c, respectively. Each snapshot process may be a computer process operating on the server system and designed for generating a snapshot of that portion of that storage volume which is maintained by its respective server. Accordingly, the snapshot process 22a depicted in Figure 5 may be responsible for generating a copy of that portion of storage volume 18 that is maintained by server 161. This operation is depicted, at least in part, by Figure 5 showing a page 28 and a copy of the page 29.

In operation, each of the equivalent servers 161, 162 and 163 is generally capable of acting independently. Accordingly, the snapshot processes 22a, 22b and 22c must act in a coordinated manner to create an accurate snapshot of the storage volume 18 at a particular point in time. This need for coordination arises, at least in part, from the fact that write requests may be issued from the client's 12a through 12e at any time and to any of the servers 161, 162 and 163. Accordingly, write requests will be received by individual ones of the servers 161, 162 and 163 during the time that a snapshot process has begun. To prevent a snapshot process from generating unacceptable or unexpected results, the snapshot processes 22a, 22b and 22c coordinate their operation with each other for the purposes of generating state information that is representative of the state of

the partitioned storage volume 18 at a particular point in time. Specifically, in one practice a time parameter is selected such that there is a time “T”, shortly after the issuing of the command to create a snapshot, such that all write operations for which completion is indicated to the client 12 prior to “T” are included in the snapshot, and no
5 write operations for which completion is indicated after “T” are not included in the snapshot.

To this end, each snapshot process 22a, 22b and 22c is capable of receiving a request from an administrator to create a snapshot of the storage volume 18. The snapshot process includes a coordinating process that will generate commands for
10 coordinating the activities and operation of the snapshot processes operating on other servers that are supporting the storage volume of interest to the administrator. In the example depicted in Figure 5, an administrator may issue a snapshot command to the snapshot process 22b operating on server 162. The snapshot command may request the snapshot process 22b to create a snapshot of the storage volume 18. The snapshot
15 process 22b can access the routing table 22b to determine those servers in the server group 16 that are supporting at least a portion of the data blocks within storage volume 18. The snapshot process 22b may then issue a command to each of the servers supporting a portion of the storage volume 18. In the example of Figure 5, each of the servers 161, 162 and 163 are supporting a portion of the storage volume 18.
20 Accordingly, the snapshot process 22b may issue a command to each of the snapshot

processes 22a and 22b to prepare for creating a snapshot. At the same time, the snapshot process 22b can begin itself to prepare to create a snapshot of that portion of the storage volume 18 maintained on server 162.

In one practice, shown in Figure 7, in response to receiving the command from snapshot process 22b to prepare for creating a snapshot, each of the snapshot processes 22a, 22b and 22 c, may suspend all requests received by clients impending execution. This may include write requests and read requests as well as any other requests appropriate for the application. To this end, each snapshot process 22a, 22b and 22c may include a request control process that allows the snapshot process to process requests being carried out by its respective server and suspend operation of those requests, thereby putting a hold on write operations that may change the state of the storage volume 18.

Once the snapshot process has suspended processing of requests, it may generate a reply to the coordinating snapshot process 22b indicating that the server is ready to begin taking a snapshot of the storage volume 18. Once the coordinating snapshot process 22b has received a ready signal from each of the servers 22a and 22c and has determined that it is also ready for a snapshot operation, the coordinating snapshot process 22b may issue a snapshot command to each of the appropriate servers. In response to receiving the snapshot commands, the server may activate, optionally, an archive process that generates state information that is representative of a copy of the

data blocks of volume 18 maintained by that respective server. In one practice and one embodiment, a mirror image is created, through a “copy on write” process such that the portions (pages) of the volume which have not changed since the creation of the snapshot are recorded once. That mirror image may be transferred to tape or other archival storage at a later time if desired. Such techniques are known in the art, and the technique employed may vary according to the application and as appropriate given the volume of the mirror image and other similar criteria.

Once the state information has been created, the snapshot process is terminated and the servers may release any suspended or pending requests for processing.

Figure 7 depicts one process according to the invention for generating a snapshot image of a data volume that has been partitioned across the servers 161, 162 and 163. As described more fully herein the distributed snapshot 70 depicted by Figure 7 allows a storage administrator to generate information representative of the state of the storage volume 18 at a particular point and time. The state information generated may include information such as the file structure, meta-data about the stored data, copies of the data maintained by the partitioned storage volume or copies of portions of the storage volume, or other such information. Accordingly, it will be understood that the snapshot process described herein has many applications including applications wherein information is generated about the structure of the partitioned data volume and stored for later use as well as applications wherein a complete archived copy of the

partitioned storage volume is created. The distributed snapshot process described herein may be employed in other applications and such other applications shall be understood to fall within the scope of the invention.

Figure 7 depicts a time/space diagram that shows a sequence of operations that
5 implement a snapshot request for the purpose of generating state information of a
partitioned storage volume or storage volumes. In particular, Figure 7 depicts a
multistage process 70 that creates a consistent distributed snapshot of the storage
volume. To this end, Figure 7 depicts three vertical lines to represent the three
servers, 162, 162 and 163 shown in Figure 5. Arrows 72 through 78 depict write
10 requests issued from one or more clients 12, and arrows 82-88 represent responses
from respective ones of the servers 161, 162 and 163.

As shown in Figure 7, the process 70 begins when a snapshot command is issued
from an administrator. In this case, the snapshot command is issued from the
administrator and delivered to server 162. The snapshot command is depicted as
15 arrow 90 directed to server 162. As shown in Figure 7, the snapshot process
executing on server 162 responds to the snapshot command by generating commands
for coordinating the operation of the other servers 161 and 163. The commands will
coordinate the snapshot processes executed on servers 161 and 163 and generate state

information representative of the state of the data maintained by each of the respective servers as part of the storage volume 18.

As further shown in Figure 7, the snapshot process executing on server 162 issues a prepare command 92 and 94 to each of the respective servers 161 and 163. The
5 snapshot processes operating on each of these respective servers 161 and 163 respond to the prepare command by holding pending requests received from clients prior to the arrival of the “prepare” command (e.g., request 78) and requests received subsequent to the “prepare” command (e.g., request 76).

Once requests have been held, the servers 161 and 163 reply to the server 162 that
10 issued the prepare command indicating that the respective servers 161 and 163 have suspended all pending requests. The server 162 acting as the coordinating server then issues the snapshot command to each of the servers. This is shown in Figure 7 by the arrows 98 and 100.

In response to the snapshot command, servers 161 and 163, as well as server 162,
15 create a snapshot of the portion of the data volume maintained by that respective server. The snapshot information may then be stored in a data file on each of the respective servers. In an optional practice, the snapshot processes on each of the servers 161, 162, and 163, may generate an archive copy of the data volume. The

archive copy may be transferred to a tape storage device, or some other mass storage device.

The snapshot generated will contain all of the request completed in the region 104 and none of those completed in region 110.

5 FIG. 8 depicts an alternative embodiment of a process for generating a snapshot of a storage volume. Specifically, FIG. 8 depicts a space-time diagram that shows a process 120 as it occurs over three time periods. These time periods are depicted in FIG. 8 as different shaded regions within the space-time diagram and are labeled as time periods 122, 124 and 126. Time period 122 occurs before the time at which an administrator issues a snapshot request, time period 124 occurs between the time
10 period that the snapshot request is issued and the snapshot operation begins, and time period 128 occurs after the snapshot has been created. The request for a snapshot operation is shown by the arrow 140 and different write requests are illustrated by the arrows 130 through 138. Responses to the write requests are illustrated by arrows
15 131, 133, 135, 137 and 139. As in FIG. 7, the three servers of the system 10 depicted FIG. 1 as shown by the vertical lines which are labeled server 161, 162 and 163 respectively.

The process 120 depicted by FIG. 8 illustrates the creation of a consistent distributed snapshot through the use of time stamps and synchronized system clocks.

More particularly, the process 120 illustrates that the servers 161, 162 and 163 can receive a plurality of write requests, each of which can arrive at one of the respective servers at any particular time. This is shown in FIG. 8 by the write requests 130, 132 and 136 which occur during the time period 122. As further shown in FIG. 8 write request 134 may arrive during the time period 124 and write request 138 may arrive during the time period 128. Accordingly, the process 120 depicted in FIG. 8 is designed to handle write requests that can occur before, during and after the snapshot process.

The snapshot process begins when a snapshot request 140 is received by at least one of the servers 161, 162 and 163. FIG. 8 depicts snapshot request 140 being sent from an administrator to the server 162. Upon receipt of the snapshot request 140, the snapshot process operating on the server 162 may issue "prepare" commands to the other servers that are supporting the data volume for which the snapshot is being created. The prepare command is depicted by the arrows 142 which is sent from server 162 to the servers 161 and 163. Upon receipt of the prepare command, the servers 161 and 163 as well as server 162, prepare for a snapshot. In this case, requests that are still pending at the servers are allowed to proceed and can be acknowledged as soon as they finish, as it is not necessary to hold them pending. Instead the servers 161, 162 and 163 determine the time at which each such request was processed and time stamps each of the respective requests. In the example

depicted by FIG. 8, this time stamping is done to write requests 136, 134 and 138, all of which are pending or received after the snapshot request 140 has been received by server 162. Once the coordinating server 162 receives a "ready" response from each of the servers 161 and 163, the coordinating server 162 generates a command to take a snapshot and transmits this command to the waiting servers 161 and 163. This command includes a time-stamp, which is the current time. This is illustrated in FIG. 8 by the arrows 160 and 162 that represent commands to the servers 161 and 163. When servers 161 and 163 receive this command, the servers include write requests with time stamps earlier than the time transmitted with the commands 161 and 162 in the snapshot. Write requests with time stamps later than the time stamp of the take snapshot commands 160 and 162 are not included in the generated snapshot. In the example depicted in FIG. 8 the write requests 136 and 134 are included within the generated snapshot while the write request 138 is not included within the generated snapshot. Once the snapshot information is generated, the process 120 may proceed as the process 70 described above with reference to FIG. 7.

Although FIG. 1 depicts the system as an assembly of functional block elements including a group of server systems, it will be apparent to one of ordinary skill in the art that the systems of the invention may be realized as computer programs or portions of computer programs that are capable of running on the servers to thereby configure the servers as systems according to the invention. Moreover, although FIG. 1 depicts

the group 16 as a local collection of servers, it will be apparent to those of ordinary skill in the art that this is only one embodiment, and that the invention may comprise a collection or group of servers that includes server that are physically remote from each other.

5 As discussed above, in certain embodiments, the systems of the invention may be realized as software components operating on a conventional data processing system such as a Unix workstation. In such embodiments, the system can be implemented as a C language computer program, or a computer program written in any high level language including C + + , Fortran, Java or basic. General techniques for such high
10 level programming are known, and set forth in, for example, Stephen G. Kochan, *Programming in C*, Hayden Publishing (1983).

 While the invention has been disclosed in connection with the preferred embodiments shown and described in detail, various modifications and improvements thereon will become readily apparent to those skilled in the art. Accordingly, the spirit
15 and scope of the present invention is to be limited only by the following claims.